QUEUES: CONCEPTUAL

Queues Implementation

2 min

Queues can be implemented using a linked list as the underlying data structure. The front of the queue is equivalent to the head node of a linked list and the back of the queue is equivalent to the tail node.

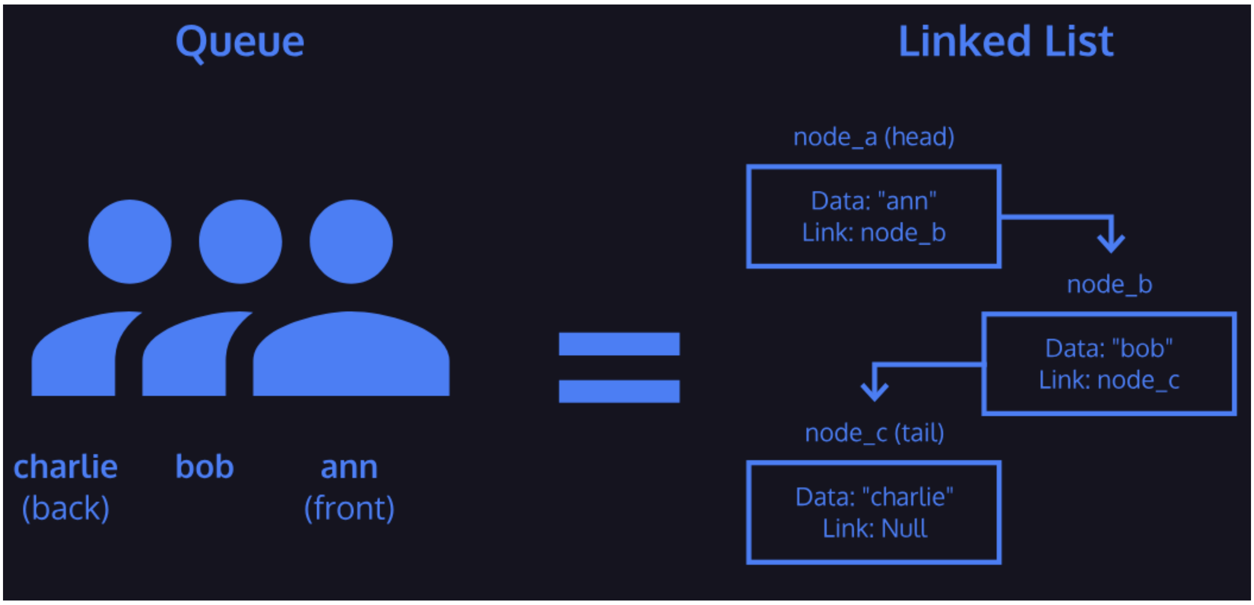
Since operations are only allowed affecting the front or back of the queue, any traversal or modification to other nodes within the linked list is disallowed. Since both ends of the queue must be accessible, a reference to both the head node and the tail node must be maintained.

One last constraint that may be placed on a queue is its length. If a queue has a limit on the amount of data that can be placed into it, it is considered a *bounded queue*.

Similar to stacks, attempting to enqueue data onto an already full queue will result in a *queue overflow*. If you attempt to dequeue data from an empty queue, it will result in a *queue underflow*.

Instructions

Why would a Linked List be a good data structure for a Queue implementation?


Queue
Linked List
charlie
(back)
bob
ann
(front)
node_a (head)
Data: "ann"
Link: node_b
node_b
Data: "bob"
Link: node_c
node_c (tail)
Data: "charlie"
Link: Null